Tutorial Seven: Simulating Gravity

Chris Agostino
Imad Pasha

March 10, 2016

1 Introduction

In lecture, we discussed how we can use programs to simulate dynamic physical situations. It is often difficult (or impossible) to analytically solve how physical bodies move through space as a function of time. One example of this is the three-body problem. With this motivation, let us begin to make a model of our solar system!

2 Gravity

In our investigation of planetary bodies, the only force that matters is that of gravity. The gravitational force between two bodies m_1 and m_2 is defined as

$$F_g = G\frac{m_1 m_2}{r_{21}^2} \tag{1}$$

where G is the gravitational constant, $6.67 \cdot 10^{-11}$ with some units. In addition to this equation, it will be useful to remember that the force is

$$F = ma \tag{2}$$

and that if acceleration is constant (which we will assume it is over short intervals), the velocity of the object is then

$$v = a \cdot t \tag{3}$$

where t is our timestep and a is determined from our force equation. Lastly our position in this linear regime we can approximate as

$$x = v \cdot t \tag{4}$$

A nontrivial fact about the gravitational force is that obeys the law of superposition. In other words, the total force on an object is just the sum of all of the individual forces on that object.

3 Writing the Code

3.1 Classes

We want to develop a class of generalized bodies which obey gravity. We want to make a class with all the necessary parameters. For this project, we are going to work in two dimensions. It is quite easy to generalize to a third dimension but 3D plotting tools are not the most fun to work with.

1. Create a body class which requires mass, position (x,y), and velocity (x,y) as its initialization parameters.
2. In addition to these initialization parameters in your init function, set up two empty arrays for x position and y position so that as the simulation progresses we can keep track of how the body moved.
3. Create a gravity method which finds the force between the two objects. Then find a way to convert that total force into its x and y components and return those values.

3.2 Simulating bodies

Now that we have our class to describe some arbitrary body. We can begin simulating how these move with time.

1. Begin by defining a simulate function which takes in a list of bodies.
2. Define your timestep (in seconds). In my implementation I use one day but test some values out. You will also want to set a number of steps. A good number would be on the order of a hundred days so you can see some sort of larger structure to the orbits. .
3. Create a for or while loop which will loop the number of steps you want.
4. In this while/for loop, Create two empty arrays, one for force in the x direction and one for force in the y direction.
5. Then, inside of the loop, create another for loop which loops over the different bodies. Inside of that loop, make two variables, $total_{fx}$,$total_{fy}$, and set them equal to zero. We will be adding to these when we measure gravity.
6. In that loop, create another for loop that loops over the different bodies. Inside of this loop, check to make sure that the two bodies are not the same body. If they are indeed different, use the gravity method to find the force between the two bodies. Add the values from the gravity function to the $total_{fx}$ and $total_{fy}$ variables from the previous step.
7. Outside of that loop, append the $total_{fx}$ and $total_{fy}$ to the arrays defined in step 4.
8. Now we have all of the forces on all of the bodies. In the while/for loop, create another for loop which we can use to extract the information from the force and bodies array.
9. In this for loop, use the information from the force array and the mass of the body to determine the accelerations in the x and y directions. Then use that and the timestep to change the velocity.
10. With the new velocity, adjust the new x and y positions and these values to our empty position arrays that we defined in the body method.
11. Now we're done with all of actual computation! Now we simply need to show our results. Use another for loop (outside of the while/for loop) to plot the position arrays we set up before.

    When you are all done, turn in your resultant plot and your code.